Developing and running an Internet crawler

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

First publication of this article on 23 December 2020

https://www.bortzmeyer.org/statistics-gemini.html

I've just developed and now runs an Internet crawler. Yes, the sort of bots that goes around and gather content. But not a Web crawler. This is for Gemini. Here are a few lessons learned.

If you don't know Gemini, there is no Wikipedia page yet, so you have to go to the official site <https://gemini.circumlunar.space/>. Basically, Gemini is born from a disenchantment : the Web is technically too complicated, it is now impossible to write a browser from scratch and because of that the competition is very limited. This technical complexity does not even serve the interests of the users : it creates distractions (such as videos automatically starting), it encourages Web sites authors to focus on the look, not on the content, and it adds a lot of opportunities for surveillance. Gemini relies on a simpler protocol, with no extensions, and a simpler format.

Today, the "geminispace" is quite small. This makes easy and reasonable to write a crawler to explore it completely. This is what I did with the Lupa <https://framagit.org/bortzmeyer/lupa> program, which is now live on one of my machines. The goal of this crawler is not to be used for a search engine (unlike the crawler used in the Gemini search engine gus.guru) but to do statistics and to survey the geminispace. You can see the results on Gemini at the URI gemini://gemini.bortzmeyer.org/software/lupa/ If you don't have a Gemini client <https://gemini.circumlunar.space/clients.html> yet, here is what it looks like, seen with the Lagrange client <https://gmi.skyjake.fi/lagrange/> :

Now, what is the current state of the geminispace ?
— 506 capsules (a Gemini capsule is a single host, a bit like a Web site) are known but Lupa could reach only 415 of them. (Gemini is quite experimental and many capsules are short-lived. Also, some links are examples and point to non-existing capsules.)
— The biggest capsule, both by the number of URIs it hosts and by the number of bytes it contains, is gemini.spam.works. Its content is mostly old files, some distributed on Usenet a long time ago. The second capsule by the number of bytes is my own gemini.bortzmeyer.org because it contains a mirror of RFCs.
— The question of certificates is touchy. Gemini suggests the use of self-signed certificates, augmented with TOFU, but not everyone agrees. Today, 17 % of the known capsules use a certificate issued by Let's Encrypt, the rest being probably self-signed.

— As it is for the Web, a same machine, with one IP address can host several "virtual hosts", several capsules. We observed 371 different IP addresses, among which 17 % use a modern protocol, IPv6. The IP address with the greatest number of "virtual hosts" is at Linode and is used for public hosting of Gemini capsules, hence its huge number of virtual hosts.
— Each capsule is identified by a domain name. Which TLD are used for these names ? The most popular is `.online` but this is because the same very common Gemini hosting service hosts many capsules under its name in `.online`, which skews the results `<https://framagit.org/bortzmeyer/lupa/-/issues/4>`. After that, the most popular TLDs are `.com`, `.org`, `.net` and `.space`.
— 64,000 URIs are known but we got only 50,000 of them. Lupa follows the robots.txt exclusion system (more on that later) which means that some capsules or some parts of capsules cannot be crawled. Also, Lupa has a limit on how many URIs to get from a single capsule.
— The most common media types (60 % of the total) is of course `text/gemini` alias "gemtext", the reference format for Gemini. Plain text comes after, with 30 %. Remember that Gemini is young and a lot of content has been produced by mirroring existing content, which is often in plain text. There are even 2 % of the URIs which are in HTML, which is surprising. Only 0.3 % are in Markdown, which should be more in the Gemini spirit than HTML.
— The Gemini protocol allows servers to tag the resources with the language they're written in. The vast majority of the resources does not use this feature, maybe because they're written in English and assume it is the world's default ? Among the tagged resources, the most common language is English, then French, then (but far after) Spanish.

Keep in mind that no crawler could explore everything. If there are capsules out here which are not linked from the capsules we know, they won't be retrieved.

What are the issues when running a crawler on the geminispace ? Since bots can create a serious load on some servers, a well-behaved bot is supposed to limit itself, and to follow the preferences expressed by the server. Unfortunately, there is no proper standard to express these preferences. The Web uses robots.txt but it is not really standardized. The original specification `<http://www.robotstxt.org/robotstxt.html>` is very simple (may be too simple), and many `robots.txt` files use one or another form of extensions, such as the `Allow:` directive or globbing. (IETF is currently trying to standardize robots.txt `<https://datatracker.ietf.org/doc/draft-koster-rep/>`.) Because of that, you can never be sure that your `robots.txt` will be interpreted as you want. This is even worse in the geminispace where it is not clear or specified if `robots.txt` apply and, if so, which variant.

Another big problem when running a bot is the variety of network problems you can encounter. The simplest case is when a server replies with some Gemini error code (51 for "resource not found", the equivalent of the famous HTTP 404) or when it rejects Gemini connections right away (TCP `RST` for instance). But many servers react in much stranger ways : timeout when connecting (no response, not even a rejection), accepting the TCP connection but no response to the start of the TLS negotiation, accepting the TLS session but no response afterwards to the Gemini request, etc. A lot of network operations can leave your bot stuck forever, or make it run in an endless loop (for instance never-ending redirections). It is very important to use defensive programming and to plan for the worst. And to set a limit to everything (size of the resources, number of resources per capsule, etc). It is a general rule of programming that a simple implementation which does 90 % of the job will be done quickly (may be 10 % of the total time of the project), but a real implementation able to handle 100 % of the job will take much longer. This is even more so when managing a crawler.